REMARKS

This paper is in response to an Office Action dated May 5, 2003 as a supplemental action regarding patent application with serial number 09/606,973. Claims 1-31 are pending and stand rejected.

The Examiner is requested to approve the accompanying replacement drawings. The changes to the drawings are to correct top margin deficiencies. Claims 28 and 31 stand rejected under 35 U.S.C. 112, second paragraph. Claim 28 has been amended to address the antecedent basis for "mission-critical server" by amending the Claim to read "mission-critical program". The amendment addresses the rejection Claim 31, which depends from Claim 28. Claims 1-31 stand rejected under 35 U.S.C 102(b) as being anticipated by Davidson, U.S. Pat. No. 5,819,093.

New Claims 32-34 have been added. New Claim 32 has support on page 14. New claims 33 and 34 have support on pages 19 and 20 in the Specification.

Applicant respectfully traverse the rejections under 35 U.S.C. 102(b) and requests reconsideration of the rejections based on the arguments presented below.

Claims 1-4, 7 stand rejected by Davidson. Davidson teaches a debugger for object oriented programs over a distributed computing system. In particular, Davidson enables application programmers to debug programs using objects on remote servers. To do so, Davidson relies on a debugger engine "dbx-engine" in the remote server. Col. 11, line 27 et seq. More particularly, the dbx-engine is started on the server using a dbxWrapperFactory object, as described with Figure 14. A dbx-engine is not multi-thread safe (MT safe) and must be created on each remote host and be instructed to attach to a server. A "wrapper server" on the remote host then creates two threads. One thread waits for the dbx-engine to terminate and the other thread waits for a message from the dbx-engine that the dbx-engine is fully started. Col. 11, lines 43-67. When one thread resumes, the other thread is destroyed and the wrapper server is notified, and the dbx-engine is created on the remote host. The new dbx-engine sets a breakpoint in the remote server to check for a match of id, host name and client address. Afterward, the debugger on the remote server is implemented and another breakpoint is set on the thread in the remote server according to a call of a trigger function, which is a function which directly calls

the function at issue implemented by the user (Col. 11, liens 20-22). After the breakpoint is set for that thread, the server is continued. Col. 12, lines 19-41. Each occurrence of the user-identified function in the Interface Definition Language (IDL), whether automatically generated or not, will cause the breakpoint to fire. Col. 12, lines 46-50. If a check determines that the code is IDL generated code, another breakpoint is set and execution is continued. Col. 12, lines 49-51.

Claim 1 as amended provides for "loading a debugger into a thread of execution of the program independent of a breakpoint; and running the debugger in the thread of execution to debug the program." Support for the amendment is found generally on page 11 and 15 of the Specification. In contradistinction, Davidson teaches a dbx-engine debugger that relies on breakpoints in the thread of execution. Further, Davidson provides a debugger that sets breakpoints in the thread of execution. Davidson neither teaches nor suggests loading a debugger into a thread of execution of a program independent of a breakpoint as claimed, and Claim 1 is allowable. Claims 2-14 depend from Claim 1 and are believed allowable with Claim 1.

Claims 15-19 and 26 stand rejected by Davidson. Claim 15 as amended provides for "halting the thread of execution independent of a breakpoint" Support for the amendment is found generally on page 11 and 15 of the Specification. Davidson teaches a dbx-engine debugger that relies on breakpoints in the thread of execution. Further, Davidson provides a debugger that sets breakpoints in the thread of execution. Davidson neither teaches nor suggests halting the thread of execution independent of a breakpoint as claimed, and Claim 15 is allowable. Claims 16-22 depend from Claim 15 and are believed allowable with Claim 15.

Regarding Claim 26, the rejection is improper. Claim 26 contains elements not addressed in the Office Action in accordance with 37 C.F.R. 1.104(b). Accordingly, a next Office Action should be non-final in nature. Moreover, Davidson fails to teach "allowing other threads of execution to continue" as taught by Claim 26. Claims 27-31 depend from Claim 26 and are believed allowable with Claim 26.

Claim 23 provides inter alia, that a "debugger converts the command into a function call to the object" unlike Claims 1-4 and 7, accordingly, the rejection of Claim 23 is improper.

Unlike Davidson, Claim 23 provides that the "debugger" performs the converting of the command. Davidson provides for a server to perform converting of a command into a function call: "The server side libraries 160, 162 convert a request from a client to a call to a user provided function 164 that implements the operation." Col. 9, lines 27-31. Accordingly, Davidson neither teaches nor suggests that a "debugger converts the command into a function cal to the object." Claim 23 is believed allowable with Claims 24-25, which depend on Claim 23.

Conclusion

Claims 1-31 are pending. Claims 32-34 have been added. The rejection to the drawings has been addressed by the enclosed replacement drawings. The rejection of Claims 28 and 31 under 35 U.S.C. 112 has been addressed by amendment to Claim 28. The rejection of each independent claim has been traversed.

The application is considered in good and proper form for allowance, and the Examiner is respectfully requested to pass this application to issue. If, in the opinion of the Examiner, a telephone conference would expedite the prosecution of the subject application, the Examiner is invited to call the undersigned attorney.

Respectfully submitted,

Jeffery J. Makeever, Reg. No. 37390
LEYDIG, VOIT & MAYER, LTD.
6815 Weaver Road, Suite 300
Rockford, Illinois 61114-8018
(815) 963-7661 (telephone)
(815) 963-7664 (facsimile)

Date: September 19, 2003

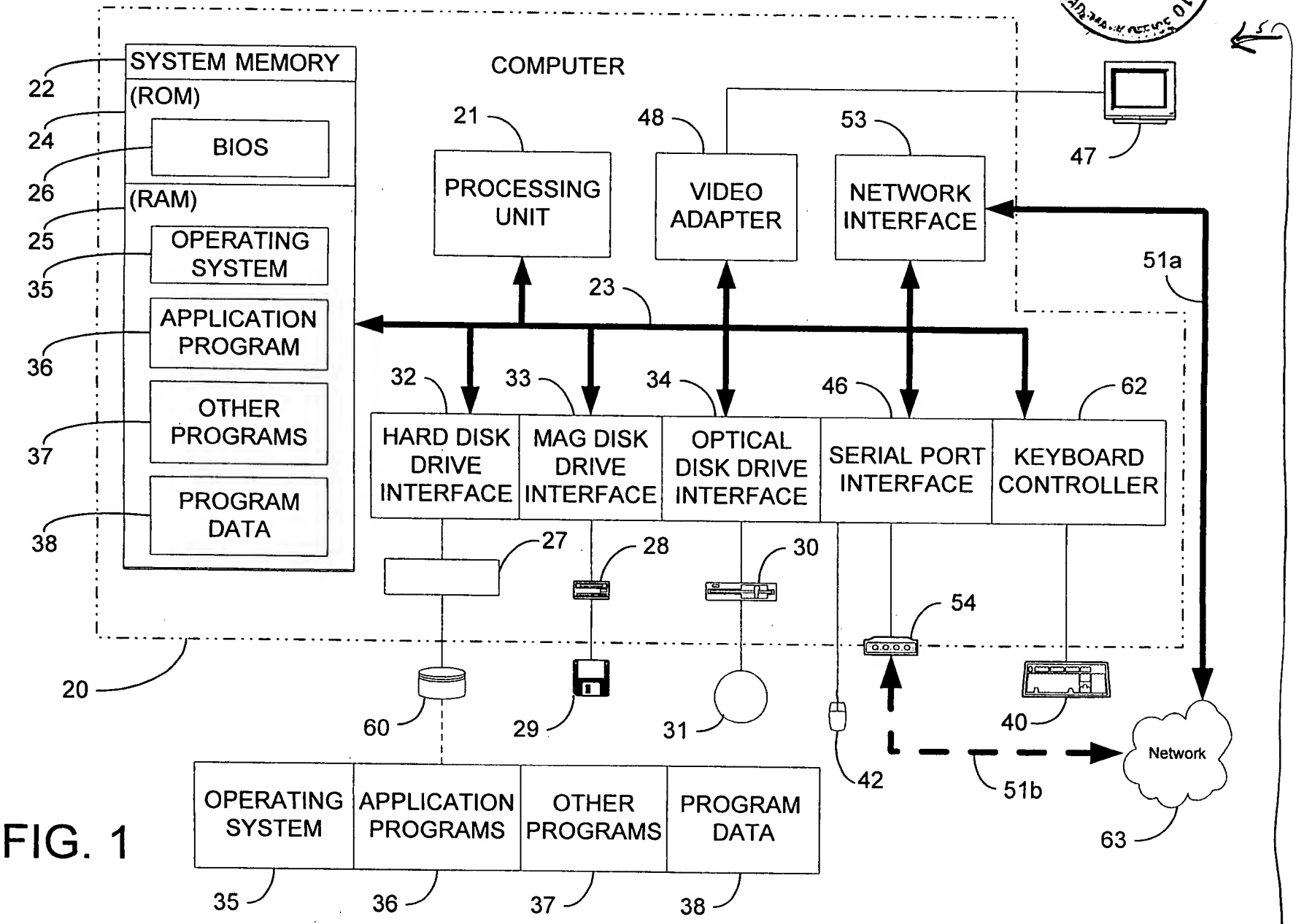
COMPUTER
SYSTEM MEMORY
(ROM)
BIOS
(RAM)
OPERATING SYSTEM
APPLICATION PROGRAM
OTHER PROGRAMS
PROGRAM DATA
PROCESSING UNIT
VIDEO ADAPTER
NETWORK INTERFACE
HARD DISK DRIVE INTERFACE
MAG DISK DRIVE INTERFACE
OPTICAL DISK DRIVE INTERFACE
SERIAL PORT INTERFACE
KEYBOARD CONTROLLER
Network
22
24
26
25
35
36
37
38
21
48
53
47
23
51a
32
33
34
46
62
27
28
30
54
20
60
29
31
40
42
51b
63
OPERATING SYSTEM
APPLICATION PROGRAMS
OTHER PROGRAMS
PROGRAM DATA
35
36
37
38

FIG. 1

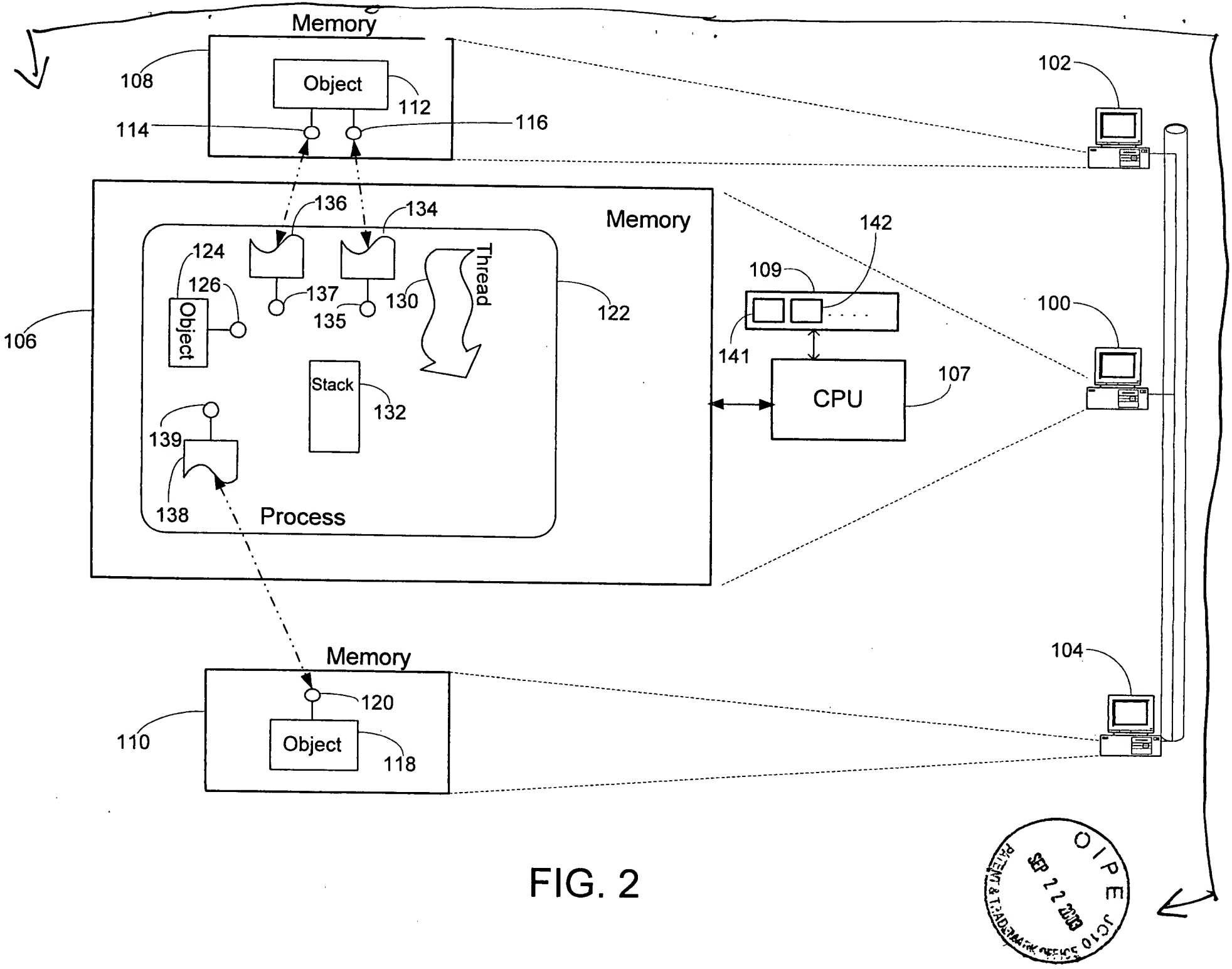

FIG. 2

OIPE
SEP 22 2008
PATENT & TRADEMARK OFFICE

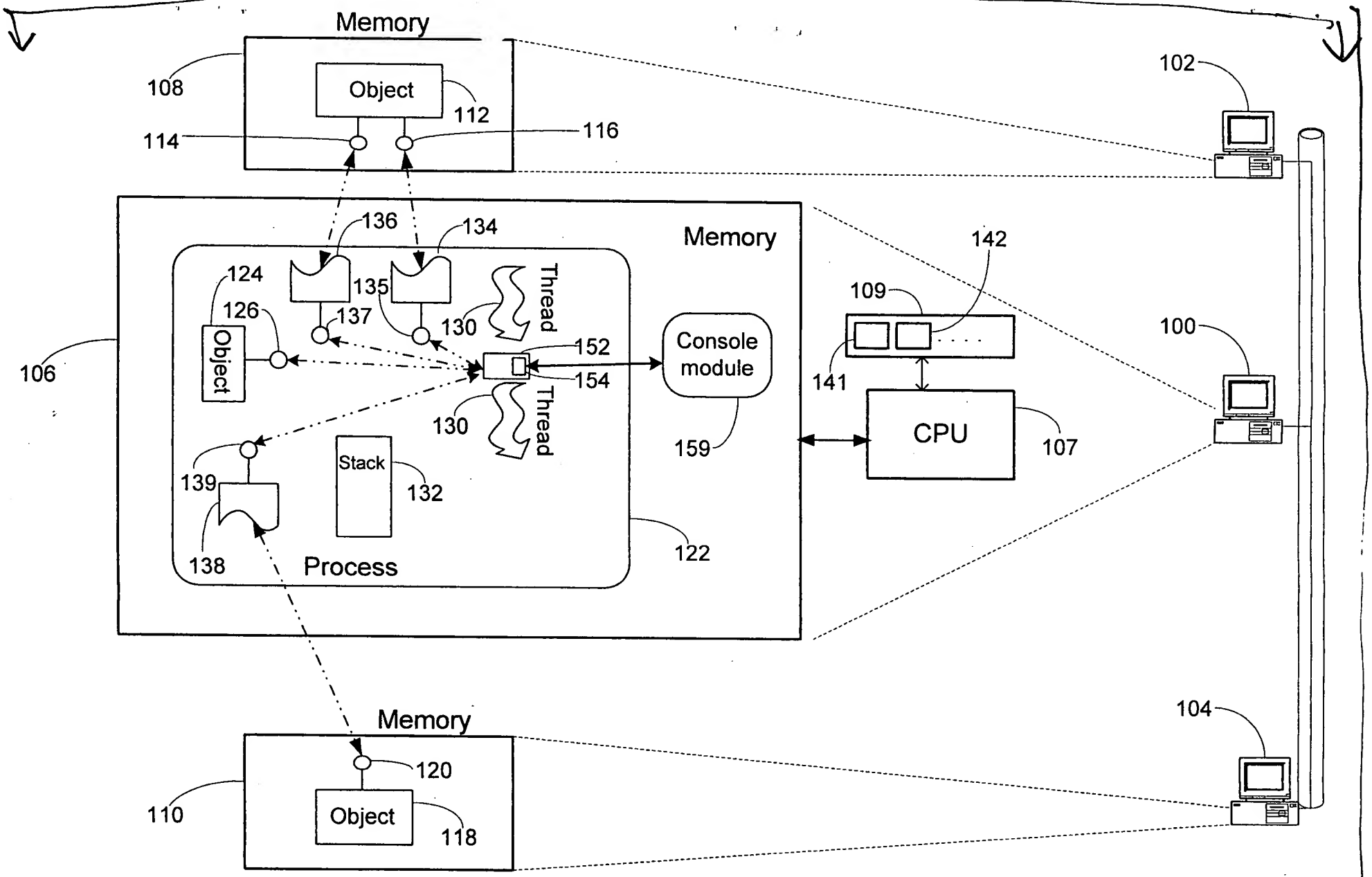

FIG. 3

OIPE
SEP 22 2008
PATENT & TRADEMARK OFFICE

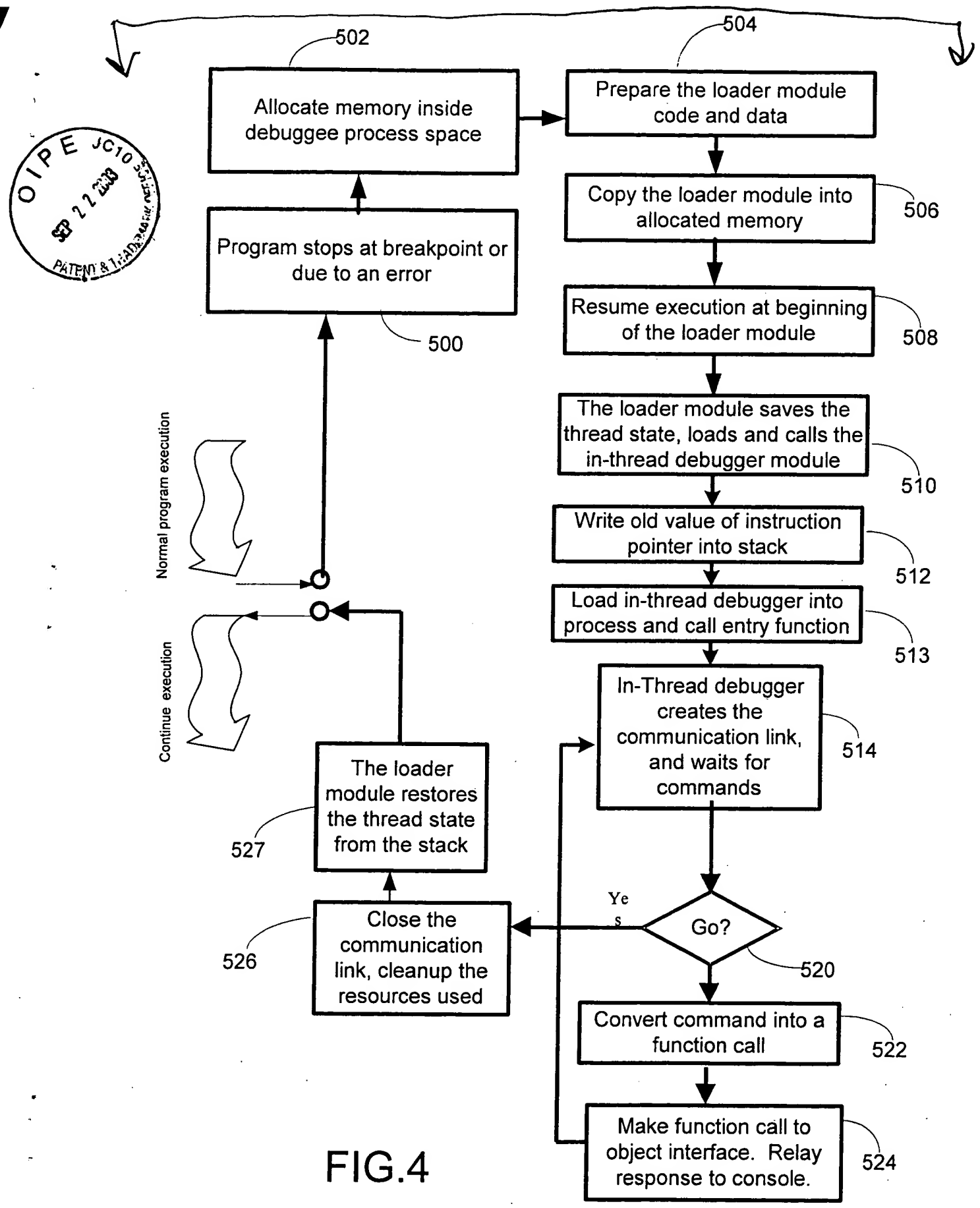
502
Allocate memory inside debuggee process space
504
Prepare the loader module code and data
Copy the loader module into allocated memory
506
Resume execution at beginning of the loader module
508
The loader module saves the thread state, loads and calls the in-thread debugger module
510
Write old value of instruction pointer into stack
512
Load in-thread debugger into process and call entry function
513
In-Thread debugger creates the communication link, and waits for commands
514
Go?
Ye
s
520
Convert command into a function call
522
Make function call to object interface. Relay response to console.
524
Close the communication link, cleanup the resources used
526
The loader module restores the thread state from the stack
527
Program stops at breakpoint or due to an error
500
Normal program execution
Continue execution

FIG.4

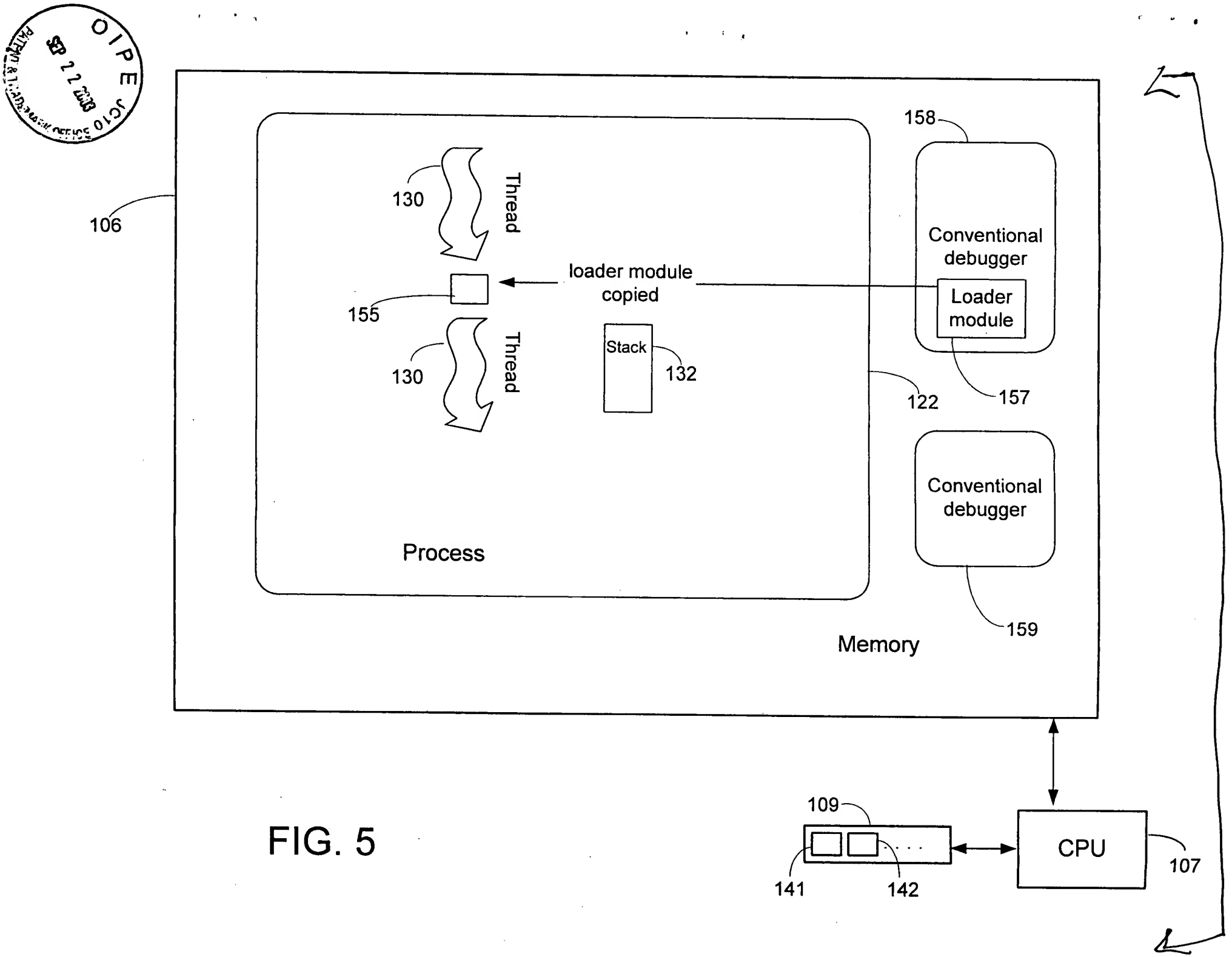
106
130
Thread
155
130
Thread
loader module
copied
Stack
132
Process
158
Conventional
debugger
Loader
module
157
122
Conventional
debugger
159
Memory
109
141
142
CPU
107

FIG. 5

OIPE JC10
22 2003
PATENT & TRADEMARK OFFICE

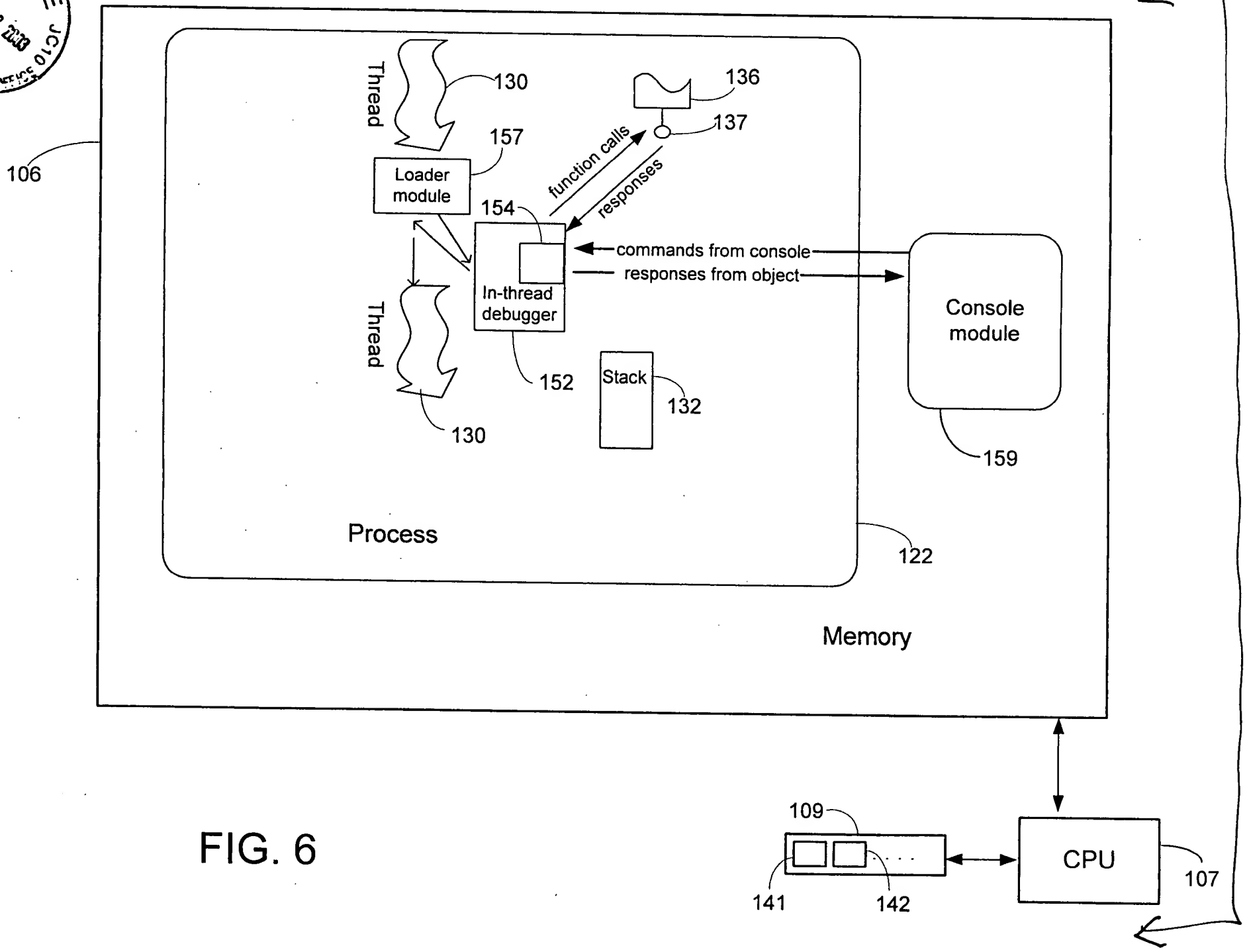

FIG. 6

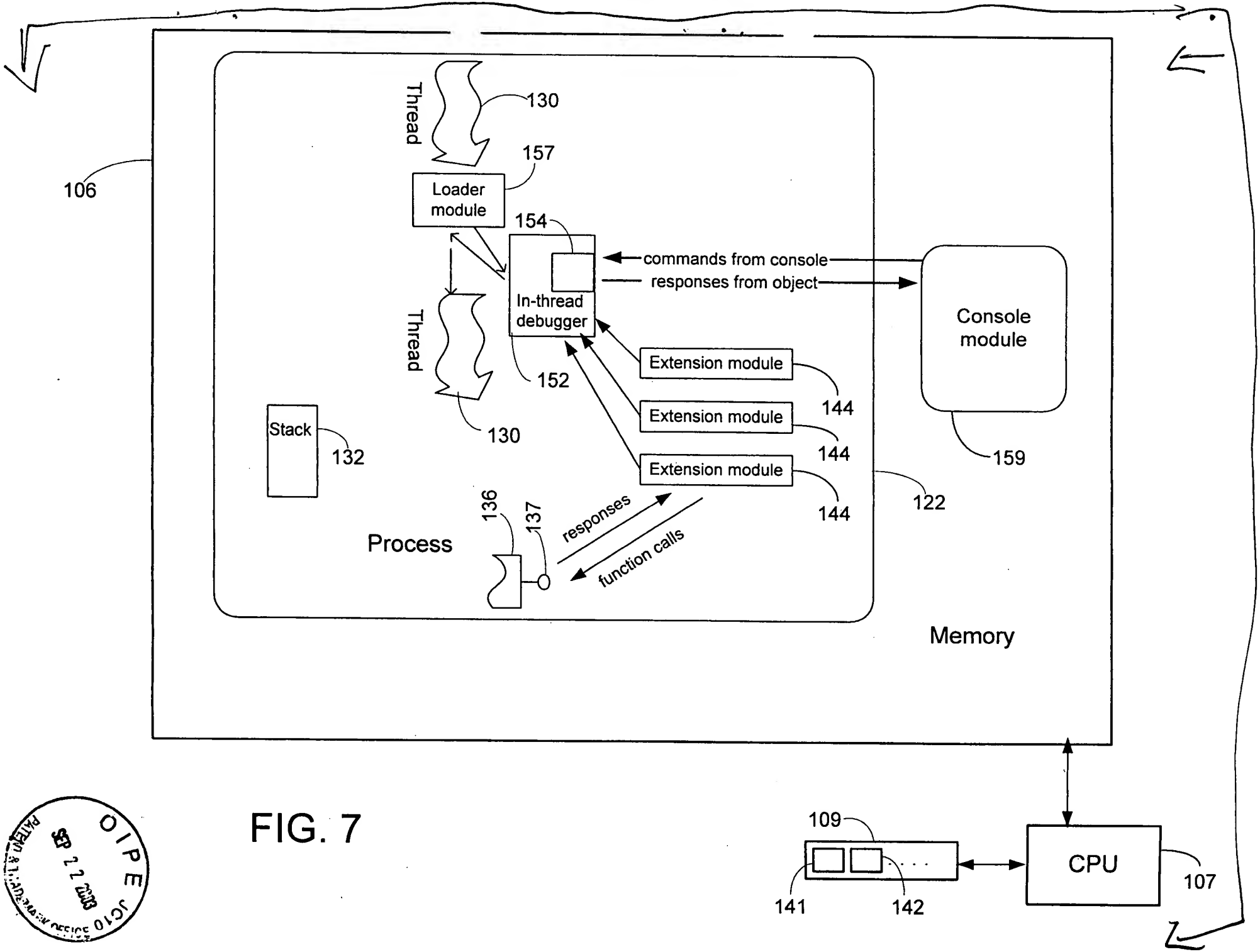

FIG. 7

PATENT & TRADEMARK OFFICE
SEP 22 2003
OIPE
JC10

OIPE JC10
SEP 22 2003
PATENT & TRADEMARK OFFICE

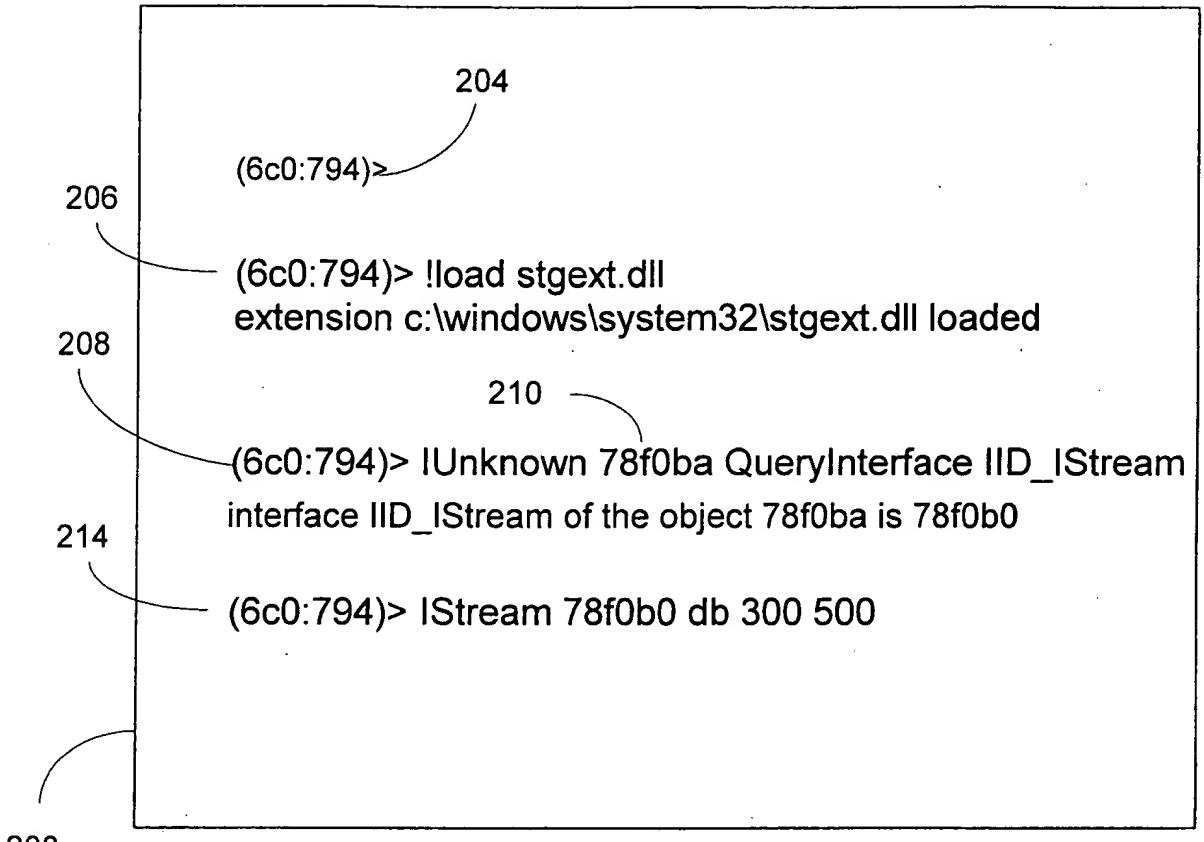

FIG. 8

OIPE JC10
SEP 22 2003
PATENT & TRADEMARK OFFICE

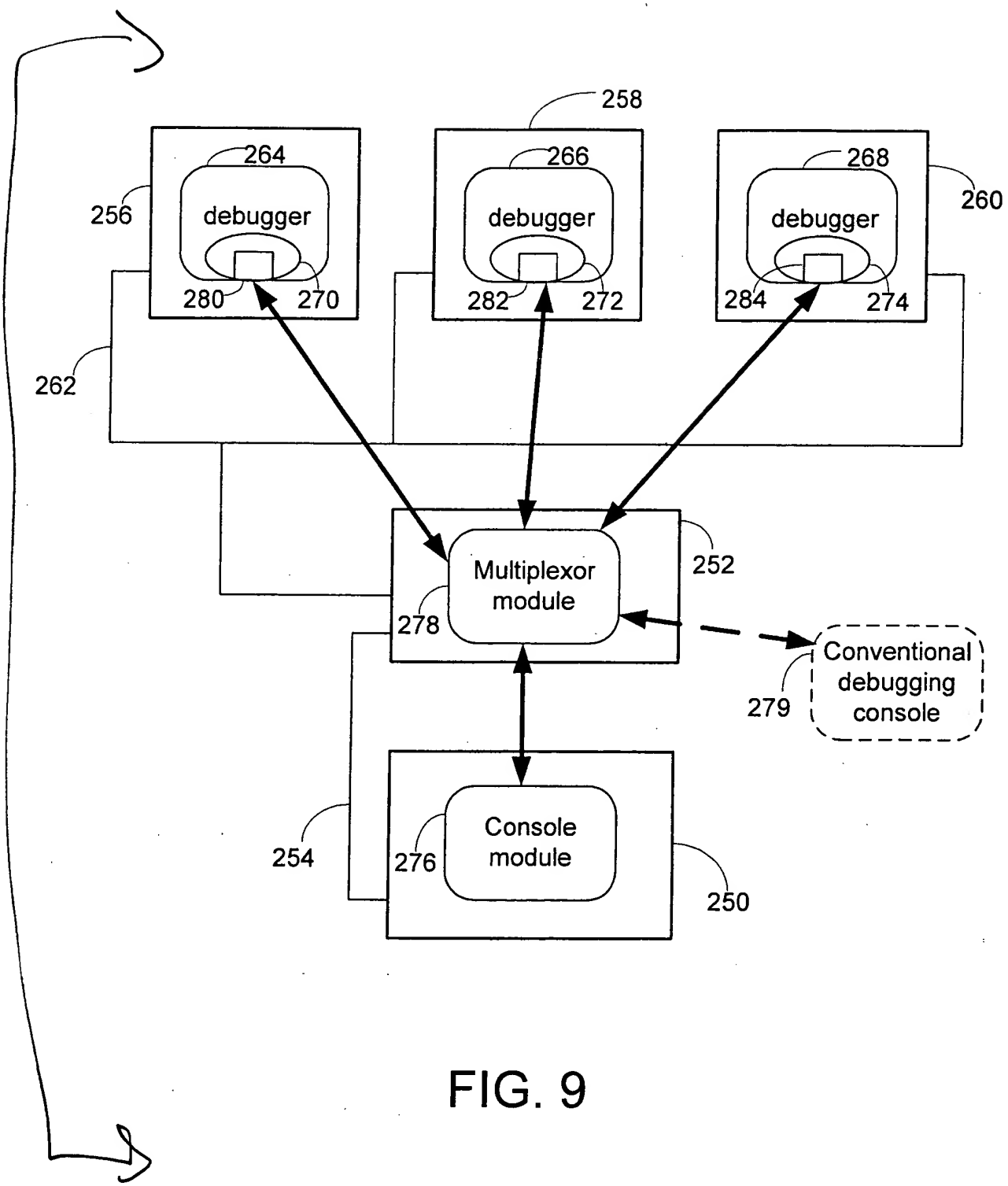

FIG. 9

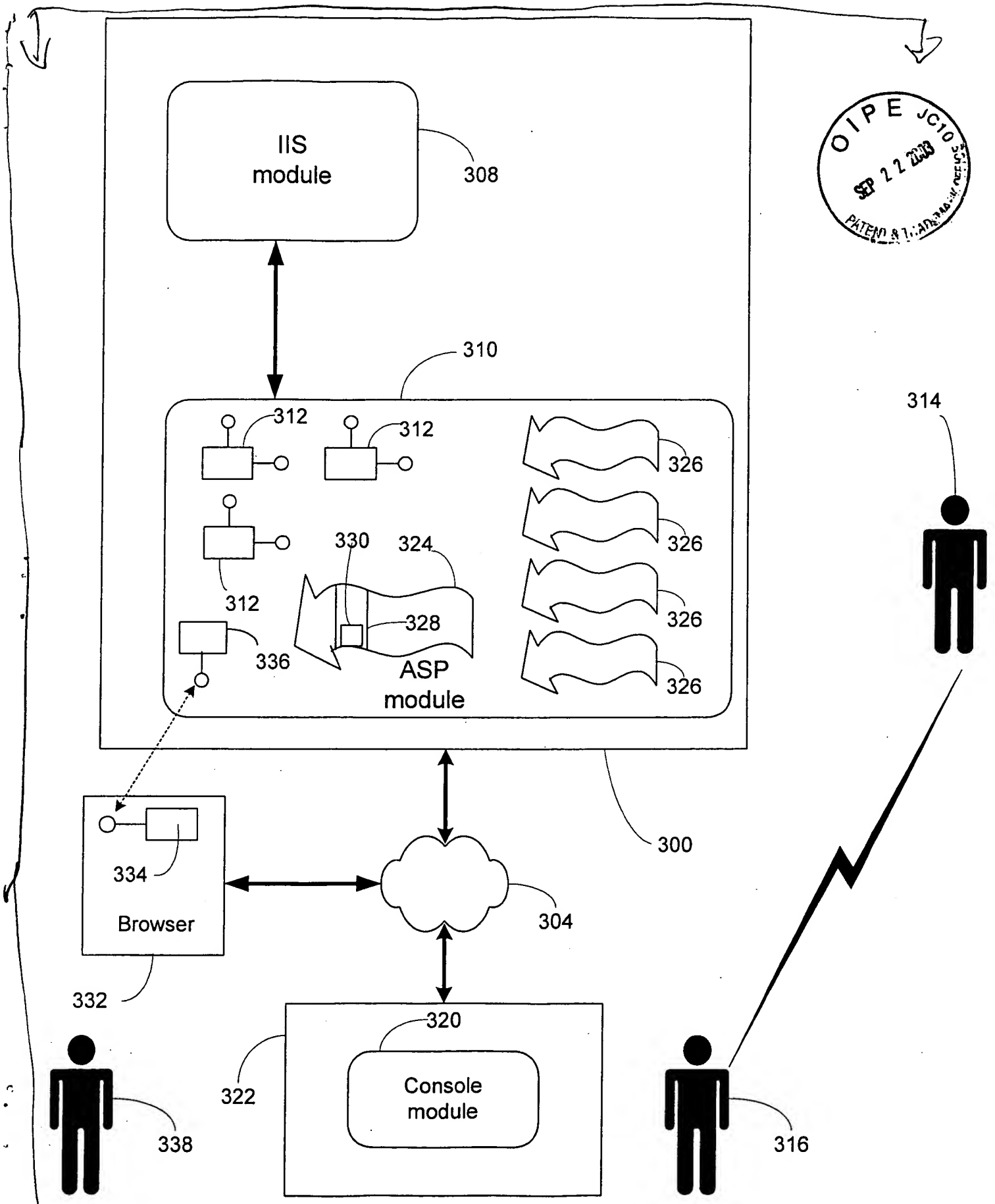
IIS module
308
310
312
312
312
326
326
326
326
330
324
328
336
ASP module
314
334
Browser
332
304
300
320
Console module
322
338
316

FIG. 10